“- “: actions

“*”: thoughts or notes

AI Robot

Logbook by Timoteo Volante

07/04Apr/25 -

-Used the FreeNove Arduino Kit Tutorial to learn about Arduino dc motors and servos.

*Challenge: Even though the design is not mine, the code will be created by me and the robot itself would also be built by me.

Robot ideas:

https://www.instructables.com/Building-a-Simple-Arduino-Robot/

chooses this: https://www.youtube.com/shorts/hWn8QDk0wE4

-Ordered 4 servos, 4 aa battery holders and an Arduino usb cable.

08/04Apr/25 -

-Tested the servo code on 180deg servo from FreeNove Arduino Kit Tutorial and found the speed for the legs: delay of 3 ms.

09/04Apr/25 -

-Tested the servo code on 360deg servo but did not work.

-Looked at codes for 360deg servo from the internet:

https://www.youtube.com/watch?app=desktop&v=OtOcuJwtPdQ

pos = 89-95 means stop, pos>95 means ccw, and pos<89 means cw.

24/05May/25 -

-Thought using a Raspberry Pi 4 and adding AI to it is more interesting than Arduino.

-Researched about power supplies for RPi 4 (battery holders and converters) and ordered heat sink and the supply.

Battery holder and converter:
https://www.amazon.co.uk/dp/B0CJ2LBY94?ref=ppx_yo2ov_dt_b_fed_asin_title

Heat sink and fan:

https://www.amazon.co.uk/dp/B07JGNF5F8?ref=ppx_yo2ov_dt_b_fed_asin_title

11/06June/25 -

Inspiration for robot design:

https://tutorials-raspberrypi.com/programming-raspberry-pi-robots-making-it-follow-the-lines/

-Ordered components from Amazon:

Microphone:
https://www.amazon.co.uk/dp/B0BHYC2MYV?ref=ppx_yo2ov_dt_b_fed_asin_title

5x battery holder:

https://www.amazon.co.uk/dp/B07MC7XC15?ref=ppx_yo2ov_dt_b_fed_asin_title

Camera module:

https://www.amazon.co.uk/dp/B076FB1KCW?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1

Short USB C to A Cable:

https://www.amazon.co.uk/dp/B0D9RGCNLN?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1

L293D Stepper Motor Drivers:

https://www.amazon.co.uk/dp/B09TK1RPSX?ref=ppx_yo2ov_dt_b_fed_asin_title

Chassis:

https://www.amazon.co.uk/dp/B07F73HY34?ref=ppx_yo2ov_dt_b_fed_asin_title

12/06June/25 -

-Used Google AI search to get an idea of how to add Open AI to RPi 4.

-Watched https://www.youtube.com/watch?v=Cj7NhuLkvdQ to update Python 3 in RPi4.

-Watched https://www.youtube.com/watch?v=lHxFFn04L10&t=271s to implement the speaking chatbot.

-Had problems with pip install “error: externally-managed-environment” so went to https://forums.raspberrypi.com/viewtopic.php?t=367098 and typed and entered

mkdir my_project

cd my_project

python -m venv --system-site-packages env

source env/bin/activate

and used pip install afterwards.

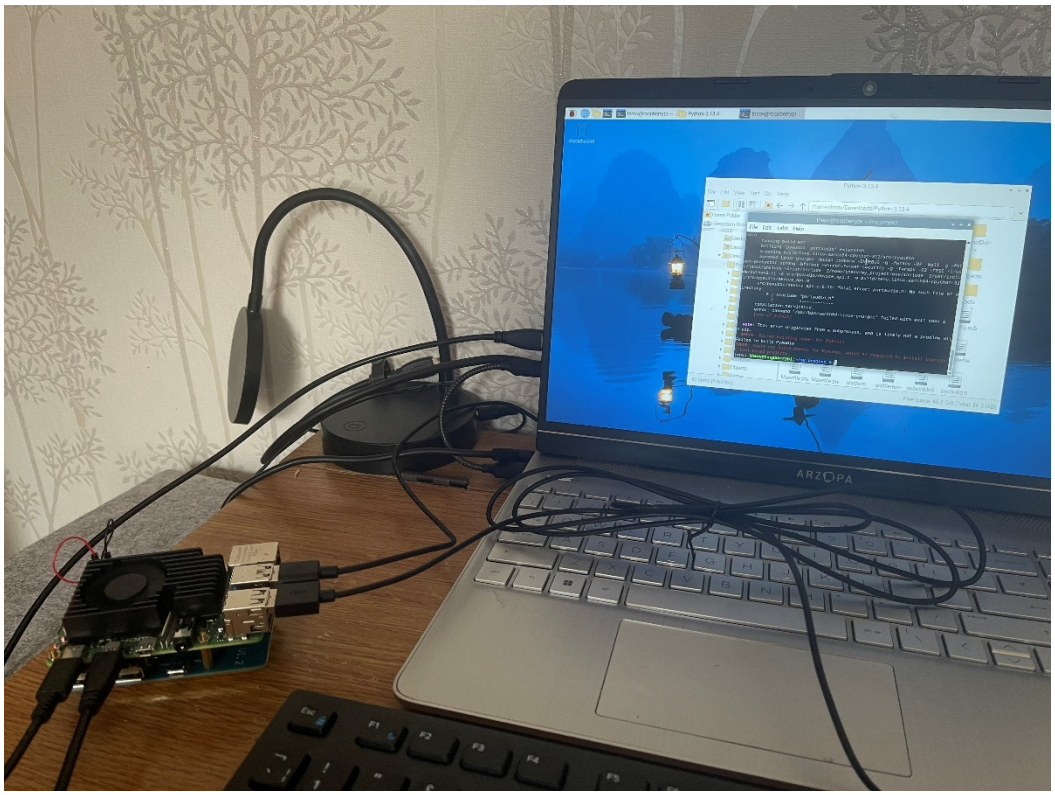
-Stopped at

1. Installing libraries/dependencies

sudo apt update
sudo apt upgrade

python3 -m pip install python-dotenv
pip3 install openai dotenv SpeechRecognition pyttsx3 gtts PyAudio
sudo apt install python3-pyaudio flac python3-espeak espeak python3-dotenv
pip uninstall dotenv
pip install python-dotenv

*Change audio to HDMI



14/06June/25 -

Pinned by @thomasthemaker

@geoffreygeo6385 11 months ago

I tried to install all the libraries. Only with PyAudio did this go wrong. Eventually, I got it to work. By the following commands:

sudo apt install libasound-dev portaudio19-dev
libportaudio2 libportaudiocpp0

After that sudo apt install ffmpeg and than run

python3 -m pip install pyaudio

Everything works great

Show less

-Deleted my_project directory and redid the installations.

-Followed the comment by Geoffrey by typing the first typing and entering the first line and then following the step 1 of the youtube description.

-Continued debugging the chatbot code.

-Ran the github version but had errors.

-Fixed the first error by removing os.getenv in

api_key = os.getenv("OPENAI_API_KEY").

-Fixed another error my installing an additional library.

https://stackoverflow.com/questions/40246437/problems-with-gst-in-python-program

16/June/25-

-Ran the script.

-Had rate limit error.

-Added some money to my OpenAI account.

-Reran the script and it worked.

-Terminated the script using Ctrl + Z.

-Tried automatically running script when RPi boots up.

Followed these sites:

https://www.instructables.com/Raspberry-Pi-Launch-Python-script-on-startup/

https://raspberrytips.com/autostart-a-program-on-boot/

https://forums.raspberrypi.com/viewtopic.php?t=267113

https://linuxconfig.org/how-to-autostart-python-script-on-raspberry-pi

To find my ip address: 192.168.0.21

-Had audio source error and fixed it by changing audio input to Pro Audio.

-Automatic script when booting worked.

Problem: Sound output for the script is stuck at hdmi when booting.

-Added this exit plan code:

```
while not done:
        for event in pygame.event.get():
                if event.type == KEYDOWN:
                        done = True
```

Where KEYDOWN = ALT

But didn't work

17/06June/25-

-Made my own script from scratch by reading the library documentations from Thomas Vu Nguyen's script and taking inspirations from the script itself.

https://pypi.org/project/openai/

20/June/25-

-Continued making my own script.

-Decided to keep most of Thomas Vu Nguyen's script but added threads, modified the listen_for_wake_word() into wake_word(), listen_and_respond() into converse(), and overall flow of code.

-Added sleep function so the ai would not input its own response, and "wait" message so the user would wait for the ai to respond.

```
                try:
                    playsound("error.mp3")
                    print("Silence found, shutting up.")
                    sleep(2)
                finally:
                    wake_word(source)

            except sr.RequestError as e:
                try:
                    print(f"Could not request results; {e}")
                    engine.say(f"Could not request results; {e}")
                    sleep(2)
                finally:
                    converse(source)

def audio_process():
    while True:
        with sr.Microphone() as source:
            wake_word(source)
            converse(source)

def main():
    audio_thread = Thread(target= audio_process)

    audio_thread.start()
    while True:
        pass

if __name__=="__main__":
    main()
```

aibot.py

```
import os, pyaudio
from threading import Thread
from openai import OpenAI
import speech_recognition as sr
from gtts import gTTS
from playsound import playsound
import numpy as np
from time import sleep

language = 'en'
client = OpenAI(api_key= '[Put your key here]')
messages = [{'role': 'developer', 'content': 'You are a helpful chatbot. Answer in less than 100 words.'}]

greetings = ["Hello, What can I do for you?",
             "How can I help?",
             "Hey, How's it going?",
             "Nice to meet you."]

wait_msg = " Please wait. I am processing your question."

r = sr.Recognizer()

def ai_response(messages):
    completion= client.chat.completions.create(
        model= 'gpt-4o',
        messages= messages,
    )
    return completion.choices[0].message.content

def wake_word(source):
    print('Listening for trigger word...')
    text = 'placeholder'
    while not ('hello' in text.lower()):
        audio = r.listen(source)
        try:
```

25/06June/2025 -

-Added memory allocation and de-allocation for the list messages.

-Added wake_for_silence() function to “sleep” until silence is found to minimise

AI responding to its own voice and forming a loop.

-Removed the break inside sr.UnknownValueError: to simply let it pass to avoid unnecessary return to wake_word() everytime he hears silence quickly.

-Added timing to return the converse() back into wake_word() when a time has elapsed enough.

aibot.py

```python
import os, pyaudio, gc
from threading import Thread
from openai import OpenAI
import speech_recognition as sr
from gtts import gTTS
from playsound import playsound
import numpy as np
from time import sleep, time

language = 'en'
client = OpenAI(api_key= 'INSERT API KEY')

greetings = ["Hello, What can I do for you?",
             "How can I help?",
             "Hey, How's it going?",
             "Nice to meet you."]

wait_msg = " Please wait. I am thinking."
r = sr.Recognizer()


def ai_response(messages):
    completion= client.chat.completions.create(
        model= 'gpt-4o',
        messages= messages,
    )
    return completion.choices[0].message.content

def wait_for_silence(source):
    while True:
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
        except  sr.UnknownValueError:
            try:
                pass
            finally:
                break
        except sr.RequestError as e:
            try:
                pass
            finally:
                break

def wake_word(source):
    print('Listening for trigger word...')
    text = 'placeholder'
    while not ('hello' in text.lower()):
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            #print(f'{text}')
        except sr.UnknownValueError:
            pass
    print('Trigger word detected.')
    greet= gTTS(text=np.random.choice(greetings),lang=language)
    greet.save('response.mp3')
    playsound('response.mp3')
    wait_for_silence(source)


def converse(source, messages, i):
    time1 = time()
    while True:
        time2 = time()
        #To allocate memory for new list once memory is de-allocated
        if i == 0:
            messages = [{'role': 'developer', 'content': 'You are a helpful chatbot
            i += 1

        print('Listening...')
        audio = r.listen(source, timeout= 5)
        try:
            text = r.recognize_google(audio)
            print(f'You said: {text}')
            messages.append({'role':'user', 'content':text})

            wait= gTTS(text= wait_msg,lang=language)
            wait.save('response.mp3')
            playsound('response.mp3')
            #Send input to OpenAI API
            print(ai_response(messages))
            tts = gTTS(ai_response(messages))
            tts.save('response.mp3')
            playsound('response.mp3')
            wait_for_silence(source)
            time1 = time()

            #To de-allocate memory
            i += 1
            if i == 6:
                del messages
                gc.collect()
                i = 0

            if not audio:
                playsound("error.mp3")

        except sr.UnknownValueError:
            diff = time2 - time1
            #print(diff)
            if time2 - time1 >= 60:
                playsound("error.mp3")
                break
            pass
```

```
        except sr.WaitTimeoutError as k:
            try:
                print("Silence detected. Shutting up")
            finally:
                break

        except sr.RequestError as e:
            try:
                print(f"Could not request results; {e}")
                engine.say(f"Could not request results; {e}")
                wait_for_silence(source)
            finally:
                pass

def audio_process():
    messages = [{'role': 'developer', 'content': 'You are a helpful chatbot. Answe
    i = 1
    while True:
        with sr.Microphone() as source:
            wake_word(source)
            converse(source, messages, i)

def main():

    audio_thread = Thread(target= audio_process)

    audio_thread.start()
    while True:
        pass

if __name__=="__main__":
    main()
```

26/06June/25-

*AI API for vision inspiration:

https://cookbook.openai.com/examples/gpt_with_vision_for_video_understanding

*Getting stream video from camera:

https://stackoverflow.com/questions/70805012/cv2-videocapture-html-live-stream

-Typed and entered:

Python3 -m pip install opencv-python

after setting up virtual environment.

And it install opencv-python

30/06June/25-

-Completed the vision thread.

-Followed a few websites to fix some small errors.

https://stackoverflow.com/questions/77284901 to fix a request error about chat completions api inputting a the jpg file.

*Note: Need to reboot or (maybe cam.close()) again camera is used to fix the error about cannot access camera resources to rerun the script.

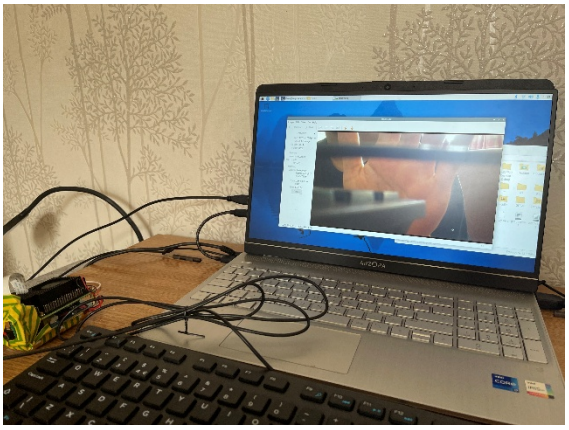
```python
def wait_for_silence(source):
    while True:
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            print(text)
        except  sr.UnknownValueError:
            try:
                print('test')
            finally:
                break
        except sr.RequestError as e:
            try:
                pass
            finally:
                break

def wake_word(source):
    print('Listening for trigger word...')
    text = 'placeholder'
    while not ('hello' in text.lower()):
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            print(f'{text}')
        except sr.UnknownValueError:
            pass
    print('Trigger word detected.')
    greet= gTTS(text=np.random.choice(greetings),lang=language)
    greet.save('response.mp3')
    playsound('response.mp3')
    wait_for_silence(source)


```

aibot.py aibot.py

```python
import os, pyaudio, gc
from threading import Thread
from openai import OpenAI
import speech_recognition as sr
from gtts import gTTS
from playsound import playsound
import numpy as np
from time import sleep, time

import cv2
import base64
from picamera2 import Picamera2


language = 'en'
client = OpenAI(api_key= '[INSERT API KEY]')

messages = [{'role': 'developer', 'content': 'Answer in less than 50 words.'}]
greetings = ["Hello, What can I do for you?",
             "How can I help?",
             "Hey, How's it going?",
             "Nice to meet you."]

wait_msg = " Please wait. I am thinking."
r = sr.Recognizer()

image_path = '/home/theov/image.jpg'


def ai_response(messages):
    completion= client.chat.completions.create(
        model= 'gpt-4.1-mini',
        messages= messages,
    )
    return completion.choices[0].message.content
```

```python
def converse(source):
    time1 = time()
    print('Listening...')
    while True:
        time2 = time()
        #To allocate memory for new list once memory is de-allocated
        audio = r.listen(source, timeout= 5)
        try:
            text = r.recognize_google(audio)
            print(f'You said: {text}')
            messages.append({'role':'user', 'content':text})

            wait= gTTS(text= wait_msg,lang=language)
            wait.save('response.mp3')
            playsound('response.mp3')
            #Send input to OpenAI API
            print(ai_response(messages))
            tts = gTTS(ai_response(messages))
            tts.save('response.mp3')
            playsound('response.mp3')
            wait_for_silence(source)
            time1 = time()

            #To de-allocate memory
            print(len(messages))
            if len(messages) == 100:
                i = len(messages)
                while len(messages) > 1:
                    messages.pop(i-1)
                    i = i - 1
                gc.collect()

            if not audio:
                playsound("error.mp3")
```

```python
        except sr.UnknownValueError:
            diff = time2 - time1
            if time2 - time1 >= 60:
                playsound("error.mp3")
                break
            pass

        except sr.WaitTimeoutError as k:
            try:
                print("Silence detected. Shutting up")
            finally:
                break

        except sr.RequestError as e:
            try:
                print(f"Could not request results; {e}")
                engine.say(f"Could not request results; {e}")
                wait_for_silence(source)
            finally:
                pass

def audio_process():
    while True:
        with sr.Microphone() as source:
            wake_word(source)
            converse(source)

def encode_image(image_path):
    with open(image_path, 'rb') as image_file:
        return base64.b64encode(image_file.read()).decode('utf-8')

def vision_process():
    cam = Picamera2()
    camera_config = cam.create_still_configuration(main={'size':(1920, 1080)}, lores={'size':(640, 480)}, display='lores')
    #picam2.start_preview(Preview.QTGL)
    cam.start()
    while True:
        sleep(10)
        print('Capturing')
        cam.capture_file('image.jpg')
        base64_image = encode_image(image_path)
        messages.append({'role':'user', 'content':[{'type':'image_url', 'image_url':{'url':f'{base64_image}'}}]})
    cam.stop()

def main():
    audio_thread = Thread(target= audio_process)
    vision_thread = Thread(target= vision_process)

    audio_thread.start()
    vision_thread.start()
    while True:
        pass

if __name__=="__main__":
    main()
```



01/07July/25-

-Replaced the de-allocation code block with the function deallocate(memory, max_len)

-Split the list into two global lists: message[] for audio and images[] for vision.

-Vision takes maximum 10 images while audio takes maximum 3 messages(1 dev msg, 1 usr msg, 1 latest image input).

*This should prevent wasteful spending of tokens (unnecessary inputs to AI).

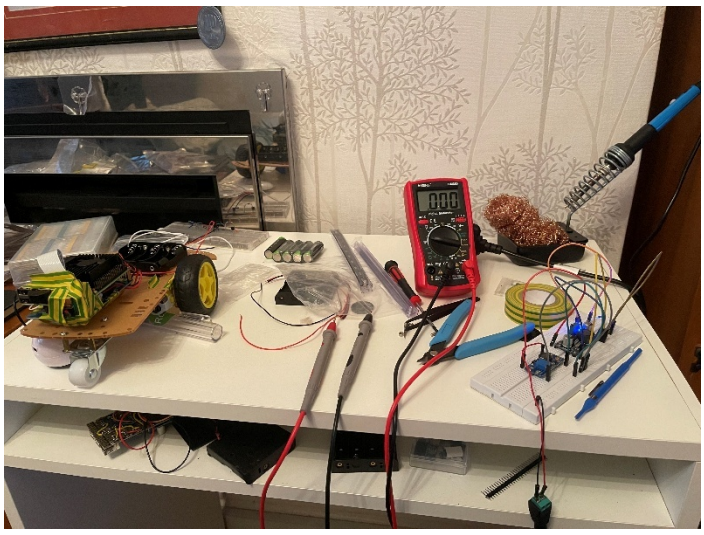
-Tried asking the AI how far away my phone is. It replied and said 1-2 feet (will use this to judge distance for driving)

-Constructed the chassis.

-Read https://lastminuteengineers.com/l293d-dc-motor-arduino-tutorial/

for the half-h drivers.

-Adjusted the regulator to output 5V from 7 - 5.5 V input.



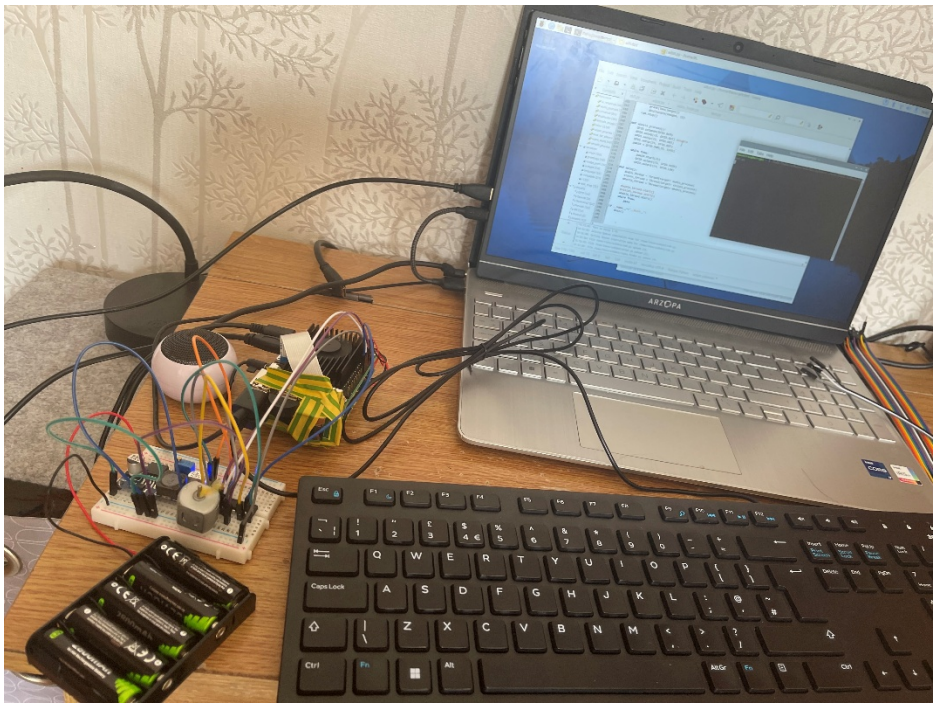
The picture above shows the built chassis on the left and the regulator on the right.

03/07July/25-

-Tested the new speaker and its volume is better than the previous one.

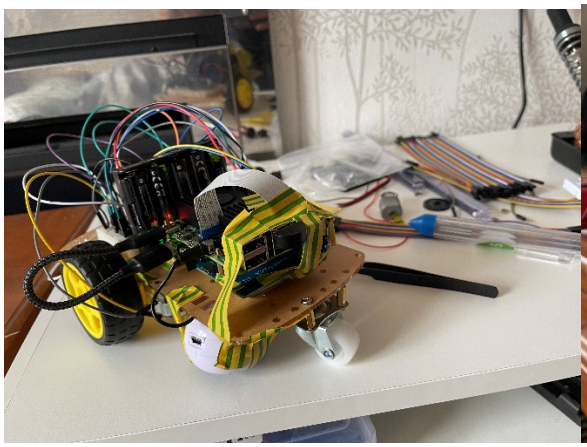
-Added the wheel control thread and its functions.

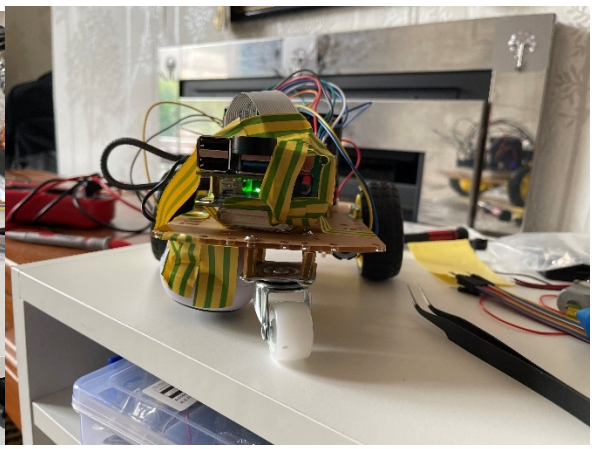
-Constructed the breadboard circuit of the motor driver and tested a 3 V dc motor.

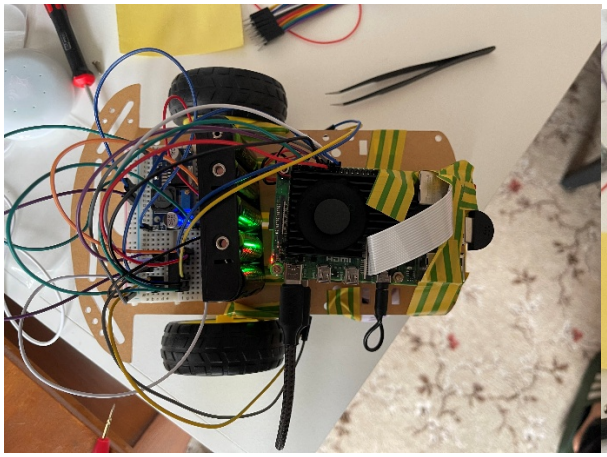


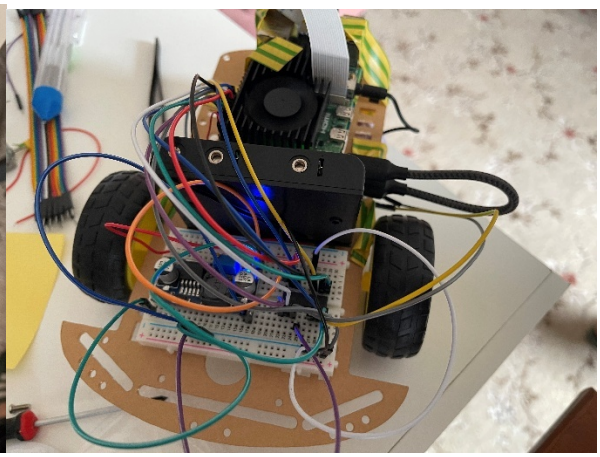
04/07July/25-

-Finished constructing the robot.









-Combined vision and wheel control thread.

-Added control_wheels() inside converse().

-Set the developer instructions such that the robot strictly only say the commands to avoid collision.

aibot.py aibot.py

```python
import os, pyaudio, gc
from threading import Thread
from openai import OpenAI
import speech_recognition as sr
from gtts import gTTS
from playsound import playsound
import numpy as np
from time import sleep, time

import cv2
import base64
from picamera2 import Picamera2, Preview

import RPi.GPIO as GPIO

######################--GLOBAL VARIABLES-#############################
language = 'en'
client = OpenAI(api_key= '[INSERT API KEY]')

messages = [{'role': 'developer', 'content': 'Answer in a human way and fewer than 50 words.'}]
images = [{'role':'developer', 'content': 'This is what you see.'}]

greetings = ["Hello, What can I do for you?",
             "How can I help?",
             "Hey, How's it going?",
             "Nice to meet you."]

wait_msg = " Please wait. I am thinking."
r = sr.Recognizer()

#Camera Initialisation
image_path = '/home/theov/image.jpg'
cam = Picamera2()
camera_config = cam.create_still_configuration(main={'size':(1920, 1080)}, lores={'size':(640, 480)}, display='lores')
cam.configure(camera_config)
#picam2.start_preview(Preview.QTGL)
cam.start()

#Pin Allocation and Wheel Control Initialisation
GPIO.setmode(GPIO.BCM)
#Left wheel
GPIO.setup(18, GPIO.OUT) #Enable
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
#Right wheel
GPIO.setup(4, GPIO.OUT) #Enable
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)

left_pwm = GPIO.PWM(18, 1000)
right_pwm = GPIO.PWM(4, 1000)

duty_cycle = 100

######################--GENERAL FUNCTIONS--#############################
def deallocate(memory, max_len):
    if len(memory) == max_len:
        i = len(memory)
        while len(memory) > 1:
            memory.pop(i-1)
            i = i - 1
        gc.collect()

def ai_response(messages):
    completion= client.chat.completions.create(
        model= 'gpt-4.1',
        messages= messages,
    )
    return completion.choices[0].message.content

```

```
######################--AUDIO FUNCTIONS--##############################
def wait_for_silence(source):
    while True:
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            print(text)
        except  sr.UnknownValueError:
            try:
                #print('test')
                pass
            finally:
                break
        except sr.RequestError as e:
            try:
                pass
            finally:
                break

def wake_word(source):
    print('Listening for trigger word...')
    text = 'placeholder'
    while not ('hello' in text.lower()):
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            print(f'{text}')
        except sr.UnknownValueError:
            pass
    #print('Trigger word detected.')
    greet= gTTS(text=np.random.choice(greetings),lang=language)
    greet.save('response.mp3')
    playsound('response.mp3')
    wait_for_silence(source)

def converse(source):
    time1 = time()
    print('Listening...')
    while True:
        time2 = time()
        #To allocate memory for new list once memory is de-allocated
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            #print(f'You said: {text}')
            messages.append({'role':'user', 'content':text})
            messages.append(images[len(images)-1])

            wait= gTTS(text= wait_msg,lang=language)
            wait.save('response.mp3')
            playsound('response.mp3')

            #Send input to OpenAI API
            #print(ai_response(messages))
            tts = gTTS(ai_response(messages))
            tts.save('response.mp3')
            playsound('response.mp3')
            wait_for_silence(source)
            time1 = time()

            #To de-allocate memory
            deallocate(messages, 3)

            if not audio:
                playsound("error.mp3")

        except sr.UnknownValueError:
            diff = time2 - time1
            if diff >= 60:
                playsound("error.mp3")
                break
            pass

        except sr.WaitTimeoutError:
            try:
                print("Silence detected. Shutting up")
            finally:
                break

        except sr.RequestError as e:
            try:
                print(f"Could not request results; {e}")
                engine.say(f"Could not request results; {e}")
                wait_for_silence(source)
            finally:
                pass

######################--VISION FUNCTIONS--##############################
def encode_image(image_path):
    with open(image_path, 'rb') as image_file:
        return base64.b64encode(image_file.read()).decode('utf-8')

######################--WHEEL CONTROL FUNCTIONS--##############################
def forward_left():
    left_pwm.start(duty_cycle)
    GPIO.output(23, GPIO.HIGH)
    GPIO.output(24, GPIO.LOW)

def reverse_left():
    left_pwm.start(duty_cycle)
    GPIO.output(23, GPIO.LOW)
    GPIO.output(24, GPIO.HIGH)

def stop_left():
```

```
        left_pwm.start(0)
        GPIO.output(23, GPIO.LOW)
        GPIO.output(24, GPIO.LOW)

    def forward_right():
        right_pwm.start(duty_cycle)
        GPIO.output(17, GPIO.HIGH)
        GPIO.output(27, GPIO.LOW)

    def reverse_right():
        right_pwm.start(duty_cycle)
        GPIO.output(17, GPIO.LOW)
        GPIO.output(27, GPIO.HIGH)

    def stop_right():
        right_pwm.start(0)
        GPIO.output(17, GPIO.LOW)
        GPIO.output(27, GPIO.LOW)

    def forward():
        forward_right()
        forward_left()

    def reverse():
        reverse_right()
        reverse_left()

    def left():
        stop_right()
        forward_left()

    def right():
        forward_right()
        stop_left()
```

```
    def stop():
        stop_right()
        stop_left()


    ######################--THREAD PROCESSES/FUNCTIONS--##############################
    def audio_process():
        while True:
            with sr.Microphone() as source:
                wake_word(source)
                converse(source)

    def vision_wheels_process():
        while True:
            sleep(10)
            #print('Capturing...')
            cam.capture_file('image.jpg')
            base64_image = encode_image(image_path)
            images.append({'role':'user', 'content':[{'type':'image_url', 'image_url':{'url':f'{base64_image}'}}]})

            control_wheels(base64_image)
            #To de-allocate memory
            #print(len(images))
            deallocate(images, 10)
        cam.stop()


    def control_wheels(base64_image):
        wheel_control_images = [{'role':'user', 'content':[{'type':'text', 'text': 'You are an exploring robot and you move by 24 inches everytime you s
        text= ai_response(wheel_control_images)
        #print(text)
        '''tts = gTTS(text)
        tts.save('response.mp3')
        playsound('response.mp3')'''
```

```
 wheel_control_images = [{'role':'user', 'content':
[{'type':'text', 'text': 'You are an exploring robot and you move by 24 inches everytime you say a command.
By judging the distance and from your 24 inch movement, avoid collision by strictly only saying either "forward", "reverse",
"turn left", or "turn right".'},
{'type':'image_url', 'image_url':{'url':f'{base64_image}'}}]}]
```

```
        playsound('response.mp3')'''
        if 'forward' in text.lower():
            forward()
            sleep(1)
            stop()
        elif 'reverse' in text.lower():
            reverse()
            sleep(1)
            stop()
        elif 'right' in text.lower():
            right()
            sleep(1)
            stop()
        elif 'left' in text.lower():
            left()
            sleep(1)
            stop()
        elif 'stop' in text.lower():
            stop()

    def main():
        audio_thread = Thread(target= audio_process)
        vision_wheels_thread = Thread(target= vision_wheels_process)

        sleep(40)
        audio_thread.start()
        vision_wheels_thread.start()
        while True:
            pass

    if __name__=="__main__":
        main()
```

07/07July/25-

-Modified the code such that it only adds one image at a time in the messages list to prevent rate limit errors and save tokens.

-Implemented the solution to mute the microphone when the AI is speaking to prevent self-responding.

-Recorded the 2nd demonstration.

```
import os, pyaudio, io
from threading import Thread
from openai import OpenAI
import speech_recognition as sr
from gtts import gTTS
from playsound import playsound
import RPi.GPIO as GPIO
from time import sleep, time
import subprocess

import cv2
import base64
from picamera2 import Picamera2, Preview

import RPi.GPIO as GPIO

#########################--GLOBAL VARIABLES--#############################
language = 'en'
client = OpenAI(api_key= '[INSERT API KEY]')

messages = [{'role': 'developer', 'content': 'You move if inches by saying one of the commands at a time:
                forward, reverse, left, or right, by you, not the user. You are 4 inches tall and 3 inches wide.
                Explore and avoid colliding by judging the distance, but first, respond to the user messages concisely.'}]

images = [{'role': 'developer', 'content': 'Filler element.'}]
greetings = ["Hello, What can I do for you?",
             "How can I help?",
             "Hey, How's it going?",
             "What's want you?"]

wait_msg = " Please wait. I am thinking."
r = sr.Recognizer()

#To mute and unmute the microphone
remute_command = ["amixer", "sset", "Capture", "nocap"]
unmute_command = ["amixer", "sset", "Capture", "cap"]

#Camera Initialization
image_path = "/home/thorw/image.jpg"
cam = Picamera2()
camera_config = cam.create_still_configuration(main={"size":(1920, 1080)}, lores={"size":(640, 480)}, display="lores")
cam.configure(camera_config)
#cam2.start_preview(Preview.QTGL)

#Pin Allocation and Wheel Control Initialization
GPIO.setmode(GPIO.BCM)
#Left wheel
GPIO.setup(16, GPIO.OUT) #Enable
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
#Right wheel
GPIO.setup(6, GPIO.OUT) #Enable
GPIO.setup(13, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)

left_pwm = GPIO.PWM(16, 1000)
right_pwm = GPIO.PWM(6, 1000)

duty_cycle = 100

#########################--GENERAL FUNCTIONS--#############################
def deallocate(memory, max_len):
    if len(memory) >= max_len:
        i = len(memory)
        while len(memory) > 1:
            memory.pop(i-1)
            i = i - 1
        gc.collect()

def ai_response(messages):
    completion = client.chat.completions.create(
        model="gpt-4.1",
        messages= messages,
    )
    return completion.choices[0].message.content

#########################--AUDIO FUNCTIONS--##############################
def say(text):
    tts = gTTS(text)
    tts.save('response.mp3')
    playsound('response.mp3')

def wait_for_silence(source):
    while True:
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            print(text)
        except  sr.UnknownValueError:
            try:
                #print('test')
                pass
            finally:
                break
        except sr.RequestError as e:
            try:
                pass
            finally:
                break

def wake_word(source):
    print('Listening for trigger word...')
    text = 'placeholder'
    while not ('hello' in text.lower()):
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            print(f'{text}')
        except sr.UnknownValueError:
            pass
    #print('Trigger word detected.')
    subprocess.Popen(mute_command) #Mute the mic to prevent AI self-responding(responding to its own voice).
    greet= gTTS(text=rg.random.choice(greetings),lang=language)
    greet.save('response.mp3')
    playsound('response.mp3')
    sleep(1)
    subprocess.Popen(unmute_command)

def conversation(source):
    timer = time()
    print('Listening...')
    while True:
        timer2 = time()
        #To allocate memory for new list once memory is de-allocated
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            #print(f'You said: {text}')

            #Append the image to the messages list.
            messages.append({image[len(images)-1]})
            #Append the response to the messages list.
            messages.append({'role':'user', 'content':text})

            subprocess.Popen(mute_command)
            say('wait_msg')
            sleep(1)
            subprocess.Popen(unmute_command)

            #Send input to OpenAI API
            #print(ai_response(messages))
            subprocess.Popen(mute_command)
            say(ai_response(messages))
            sleep(1)
            subprocess.Popen(unmute_command)

            #Waits until the AI stop talking to prevent self responding.
            wait_for_silence(source)

            #Makes AI control the wheels.
            move(ai_response(messages))

            timer = time()

            #Pop the image out the messages list to prevent rate limit errors and save tokens.
            messages.pop(len(messages)-1)

            #To de-allocate memory
            deallocate(messages, 100)

            if not audio:
                playsound("error.mp3")

        except sr.UnknownValueError:
            #Goes back to sleep after some time not receiving audio input.
            diff = time2 - timer
            if diff >= 1800: #30 minutes
                playsound("error.mp3")
                break

            pass

        except sr.WaitTimeoutError:
            try:
                print("Silence detected. Shutting up")
            finally:
                break

        except sr.RequestError as e:
            try:
                print(f"Could not request results; {e}")

                subprocess.Popen(mute_command)
                engine.say(f"Could not request results; {e}")
                sleep(1)
                subprocess.Popen(unmute_command)
            finally:
                pass

#########################--VISION FUNCTIONS--#############################
def encode_image(image_path):
    with open(image_path, "rb") as image_file:
        return base64.b64encode(image_file.read()).decode("utf-8")

#########################--WHEEL CONTROL FUNCTIONS--#######################
def forward_right():
    left_pwm.start(duty_cycle)
    GPIO.output(6, GPIO.HIGH)
    GPIO.output(12, GPIO.LOW)

def reverse_right():
    left_pwm.start(duty_cycle)
    GPIO.output(13, GPIO.LOW)
    GPIO.output(12, GPIO.HIGH)

def stop_right():
    left_pwm.start(0)
    GPIO.output(13, GPIO.LOW)
    GPIO.output(12, GPIO.LOW)

def forward_left():
    right_pwm.start(duty_cycle)
    GPIO.output(17, GPIO.HIGH)
    GPIO.output(27, GPIO.LOW)

def reverse_left():
    right_pwm.start(duty_cycle)
    GPIO.output(17, GPIO.LOW)
    GPIO.output(27, GPIO.HIGH)

def stop_left():
    right_pwm.start(0)
    GPIO.output(17, GPIO.LOW)
    GPIO.output(27, GPIO.LOW)

def forward():
    forward_right()
    forward_left()

def reverse():
    reverse_right()
    reverse_left()

def left():
    forward_left()
    reverse_right()

def right():
    forward_left()
    stop_right()

def stop():
    stop_right()
    stop_left()

def move(text):
    if 'forward' in text.lower():
        forward()
        sleep(1)
        stop()
    elif 'reverse' in text.lower():
        reverse()
        sleep(1)
        stop()
    elif 'right' in text.lower():
        right()
        sleep(1)
        stop()
    elif 'left' in text.lower():
        left()
        sleep(1)
        stop()
    elif 'stop' in text.lower():
        stop()

#########################--THREADS--######################################
def audio_process():
    while True:
        with sr.Microphone() as source:
            wake_word(source)
            conversation(source)

def vision_process():
    cam.start()
    while True:
        sleep(5)
        #print('Capturing...')
        #Capture and save image into the images list.
        cam.capture_file("image.jpg")
        base64_image = encode_image(image_path)
        images.append({'role':'user', 'content':[{'type':'image_url', 'image_url':{'url':f'{base64_image}'}}]})

        #Append image to the messages list.
        #messages.append({image[len(images)-1]})
        #print(ai_response(messages))

        #Makes AI control the wheels.
        move(ai_response(messages))

        #Pop image out the messages list to prevent rate limit errors and save tokens.
        messages.pop(len(messages)-1)

        #To de-allocate memory
        deallocate(messages, 100)
    cam.stop()

#########################--MAIN THREAD--##################################
def main():
    audio_thread = Thread(target= audio_process)
    vision_thread = Thread(target= vision_process)

    sleep(5)
    audio_thread.start()
    vision_thread.start()
    while True:
        pass


if __name__=="__main__":
    main()
```

09/07July/25-

-Removed the sleep() to reduce latency.

-Recorded the 3rd demonstration.